Amendments to the Specification:

Please delete the paragraph beginning on page 12, line 18 and replace with the following:

Menu interface data objects are arranged into a hierarchy which is similar but necessarily equal to the hierarchy of system resource data objects. The hierarchy of menu tasks represents the most convenient and usable view of the system resources to the user. Fig. 10A, 10B, and 10C illustrates a portion of the hierarchy which exhibits the salient characteristics of the interface tool hierarchy. The menu objects [[902]] 901-909, Fig. 10A, appear together in a logical frame presentation by virtue of sharing the same search key. Fig. 10B illustrates the branch of the hierarchy that would be available for the traversal if data object 903, Fig. 10A, were selected by the user. Menu interface data objects 916-926, Fig. 10B appear together in a logical frame presentation by virtue of sharing the same search key. Objects 927-934 would appear together in a logical frame 30 presentation if object 916 were selected. If object 932 were selected, objects 935-937 would be available for selection. These are the same objects that appear on Fig. 10A in a logical frame presentation which is reached through a different selection path through the 35 hierarchy. Fig. 10C shows a continuation of the branch available through selection of menu object 926, Fig. 10B. Objects 936 and 937, Fig. 10C, lead to different branches of menus. However, because TCP/IP is available to the user for configuration on either adapter 956, 957, some menu objects 959, 960, Fig. 10C, are shared between the two branches. Menu object 906, Fig. 10A, is repeated on Fig. 10C. Its selection causes the presentation of object 939 and 940. The selection of menu object 939 results in a logical frame presentation which includes object 955 and 945. Object 955 is also contained in a logical frame presentation arrived at by the selection of object 954. Object 945 is also contained in a logical frame presentation arrived at by the selection.

Please delete the paragraph beginning on page 22, line 3 and replace with the following:

The field multi_select 324 indicates whether the user may make multiple selections from the cmd_to_list. Allowed values are "yes' and "no". The field value_index 314 is a zero-origin index indicating the default value in the sequence of values specified in the value field if the object is a ring.

Please delete the paragraph beginning on page 36, line 24 and replace with the following:

The above description is illustrated in the block diagram of the interface tool as shown in Fig. 5A.

For example, if a user wanted to perform system management tasks on the logical volume manager in the data processing system, the model used to accomplish this is shown by Fig. 5A. The menu 1A contains all logical volume jobs from which the user can select. Some of these more general tasks 'include ADD 501, SHOW 502, DELETE 503, CHANGE 505, and LIST ALL 506. In the ADD 501 model, the user goes straight into the dialogue [[14A]] 400 after selecting ADD off of the logical volume manager menu 1A. It would have a command_to_discover which would discover the defaults. Since the user has selected ADD 501, the user is not working with an existing system resource; hence no current information exists. This distinction between discovery of defaults and discovery of current settings is of no functional importance to the interface shell. The user then fills in the information into the dialogue and executes the ADD command_to_exec.

Please delete the paragraph beginning on page 37, line 9 and replace with the following:

In using the SHOW 502 task, the user wants to see the characteristics of a system resource. Therefore, the interface has to be cognizant of which instance of this class of system resources to show. Therefore, the sm_name_select object [[601]] 600 is required. For this example, the field would display to the user "name of logical volume". If the user knew the name the user wanted, the user would enter the name into the field. Since the name select object [[601]] 600 also has a list feature ~~612, 613~~, the interface uses an operating system command to list the names of logical volumes in a panel for the user to select from.

Please delete the paragraph beginning on page 37, line 34 and replace with the following:

A sm_name_select object [[601]] 600 is also used for the DELETE 503 task since the user has to indicate to the interface shell the name of the system resource to be deleted. However, the dialogue header object 400 is actually a ghost. A ghost dialogue means that, there is no I/0 to the screen. A dialogue is not presented to the user because no further information is needed. However, if the deletion of the system resource had varieties of function (e.g. delete from active use but save info in a retention file as opposed to delete completely)', then a screen dialogue would follow for solicitation of the user's wishes. There are no dialogue objects 300 associated with a ghost. The header is sufficient to execute the end command.

Please delete the paragraph beginning on page 38, line 22 and replace with the following:

In the performance of a CHANGE task, unlike the ADD task, the system resource to be changed already exists. The sm_name_select object [[601]] 600 is utilized to get the name of the object. The command_to_discover is executed, the current values of the attributes of the system resource are found, these current values and attributes are presented to the user in the dialogue, the user responses are collected, and the CHANGE command_to_execute 505 is executed.

Please delete the paragraph beginning on page 38, line 31 and replace with the following:

The sm_name_select [[601]] 600 is not required for the LIST ALL task 506 since all of the names of the instances of this type of system resource.will be used. The dialogue is a ghost dialogue since the user does not need to give any further instructions with this command. Therefore, the LIST ALL command_to_execute is executed immediately and the user receives output to the screen. A ghost dialogue typically infers that there is no I/O to the screen for interaction with the user. However, the dialogue object 400 is still required since the command to execute 408 resides within this structure.

Please delete the paragraph beginning on page 39, line 8 and replace with the following:

A more complex example is shown in Fig. 5B with respect to the interface tool for managing printers or other physical devices in the system. To add a printer, a name select object [[601]] 600 is required since, unlike logical volumes, all printers are not the same; they are diverse types. Therefore which dialogue object 400 is to be used is not known until the type is specified in the name select [[601]] 600 facility. For example, to add a type A printer, one set of questions would be required in the dialogue, while if a type B printer were to be added another set of questions would be required in a dialogue. Therefore, to reach the correct dialogue object 400, the name_select_object [[601]] 600 would ask the user the type of printer. There are different ways in which the name 604 in sm_name_select is used. The string indicated in the next_id 603 (Figure 6) could be the actual search key for the dialogue header 400, in which case the name value is not important. However, the name may be needed to find the next dialogue header 400. For example, if a user did not know what type of printer they wanted to deal with, the user could select LIST to get a list of all supported printers. In this case, the interface takes the "type" to find the appropriate dialogue object 400. However, to perform a CHANGE 505 job, instead of changing a supported printer type, the user wants to change characteristics of an instance of a type of system resource, e.g. printer. The name of the instance might have been specified by the user and would

then be very specific to the user's machine. It would not necessarily denote a type. If the user could not remember the machine specific name, the user would want to list the possible names. However, the list returned would not have the type names, but would have user specific names which the interface shell cannot understand in order to find the correct dialogue header object. Since these user specific names are not universal, an interface tool can not provide any predefined dialogue objects 400 for each of these user specific names. Instead, within the sm_name_select object there is a command_to_classify 614 which correlates a user specific name to a printer type. The comrnand_to_classify then executes and concatenates the name with the search field to find the correct corresponding dialogue object 400.

Please delete the paragraph beginning on page 40, line 18 and replace with the following:

For performing a SHOW task 502 for printers, the sm_name_select [[602]] 600 is used to specify the name of the printer instance about which the user desires information. Since there is no further information that needs to be gathered, the SHOW command is executed directly, using the name selected, and the output is shown on the screen.

Please delete the paragraph beginning on page 43, line 27 and replace with the following:

This is illustrated with reference to Fig. 7 and objects 701-706. A top menu 701 is shown where one of its fields contains the actual text 205 displayed as "DEVICES" 205. If command to execute, and wants to use the interface tool to facilitate the entry of parameters and options, then the interface tool can be invoked with the command name as a parameter, which is the id of the dialogue header interface object which executes that command. This second approach is referred to as the fast path method. All objects, by virtue of having an id, can be reached by this fast path method by the user passing the id as a parameter in the invocation of the interface tool. The interface tool then searches for an object with the id of the parameter sent. The interface tool first searches for a dialog header object with the given id. If found, the corresponding dialogue interface objects are sent to the screen library and presented to the user as in step 814, Fig. 8, and processing continues as described above. If the dialogue header is not found, the interface tool searches for a name selector object with the given id. If a dialogue is preceded by a name selector, then the command name is the id of the name selector rather the dialog header. If a name selector object is found, processing continues with step 808 as described above. Finally, the interface tool searches for a menu with the given id, and if found, processing continues with step 801 as described 20 above. Since some commands are represented by more than one dialogue, passing the command name as a parameter can not be taken as an unambiguous reference to a single dialogue. Therefore, a command

name is the id of the lowest object in the hierarchy which represents all significant instances of the command. In most cases, this is a dialogue header, but could be a menu.

Please delete the paragraph beginning on page 45, line 16 and replace with the following:

As shown in Fig. 1B the ASL layer 20 is used to present the information to the user and collect user responses. The ASL layer is a screen library layer which protects the interface shell from dealing with differences in text or graphic libraries 21, 22, 23. Several presentation libraries 22, 23 can be interchanged to produce different presentation styles. The target graphic library may be chosen by the user at initialization time or may be based on an external environment variable. These other support libraries may include presenting graphics and text to the user. Regardless of the presentation interface 20 used, the same objects within the object data manager, database 30 are used. The data objects within the database are defined only once. Each graphics library can then use its own symbols to present the information from the objects to the user. These interchangeable graphic interfaces merely present the same information from the interface tool in different visual ways.